

A MULTIMEDIA ANIMATED COMPARISON
OF CONVEX HULL ALGORITHM

By

CHAOPANG FAN

Bachelor of Science

Shantou University

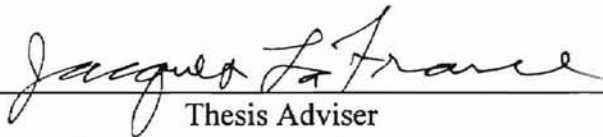
Guangdong, P.R.China

1990

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 1998

A MULTIMEDIA ANIMATED COMPARISON
OF CONVEX HULL ALGORITHM

Thesis Approved:


Thesis Adviser






Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my major advisor, Dr. Jacques LaFrance, who introduced me to this interesting field, and constantly gave me intelligent guidance and enthusiastic suggestions throughout this study. My appreciation extends to other committee members, Dr. John P. Chandler and Dr. G. E. Hedrick, for their helpful advisement and suggestions.

In addition, I wish to express my appreciation to all of my friends who gave me assistance and encouragement for this study. My special thanks and appreciation go to my mother, father and brother whose encouragement and understanding were invaluable throughout this study.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION.	1
1.1 Problem Statement	2
1.2 Organization of This Thesis	3
II. COMPUTER ANIMATION AND ALGORITHM ANIMATION	4
2.1 Computer Animation	4
2.2 Algorithm Animation	5
2.2.1 General Animation Systems	6
2.2.2 Geometric Animation Systems	7
2.3 Previous Related Work at OSU	9
III. MULTIMEDIA AND WWW TOOLS	10
3.1 HTML	10
3.2 Plug-Ins	11
3.3 Multimedia Authoring Tools	12
3.4 Card-Based or Page-Based Tools	13
3.5 Icon-Based Tools	14
3.6 Time-Based Tools	14
3.7 Object-Oriented Tools	15
IV. MACROMEDIA DIRECTOR	17
4.1 Macromedia Director Overview	17
4.2 Basic Concept	18
4.3 Event and Event Handler	19
4.4 Shockwave Movie	22
V. GEOMETRIC ALGORITHMS.	23
5.1 Convex Hull Algorithms	23
5.2 Package-Wrapping Algorithm and Analysis	24
5.3 Graham Scan Algorithm and Analysis	25
VI. SYSTEM DESIGN AND IMPLEMENTATION.	28
6.1 System Architecture	28
6.2 Line and Point Manipulation	30
6.3 Angle Calculation and Processing	38

6.4	Coordinate System Transformation.	42
6.5	P/C Script and Ancestor Script	43
6.6	Shockwave Movie Generation	45
VII.	SUMMARY, CONCLUSIONS AND SUGGESTED FUTURE WORK.	47
7.1	Summary and Conclusions	47
7.2	Suggestions For Future Work	48
	SELECTED BIBLIOGRAPHY	49

LIST OF TABLES

Table	Page
3-1 Multimedia Plug-ins and Vendors	11
4-1 Events Provided by Macromedia Director 6	20

LIST OF FIGURES

Figure	Page
4-1 Score Script Hierarchy and Handlers	21
4-2 Cast Member Script and Its Handlers	21
4-3 Movie Script and Its Handlers	22
5-1 The Convex Hull of A Set of Points	23
5-2 An Example of Package-Wrapping Algorithm..	25
5-3 An Example of Graham Scan Algorithm	27
6-1 The System Architecture	29
6-2 Flow Chart of Draw Line Function	32
6-3 Comparison of Line Implementation by <i>Paint Tool</i> and <i>Tool Palette</i>	34
6-4 Comparison of Lines in Different Quadrants	35
6-5 The Cast Member Lines and the Ranges They Represented	37
6-6 Pseudoangle and Its Range	39
6-7 The Pseudoangle Calculation	40
6-8 An Example of Angle Calculation	41
6-9 The Comparison of Two Coordinate System	42
6-10 The OOP Hierarchy of the System	44

CHAPTER I

INTRODUCTION

When students are learning a complicated new algorithm, they might read a book or a paper that explains the algorithm, or even have to pore over the code of the algorithm, which is a difficult way to learn complicated algorithms. Computer-based algorithm animation helps them learn these algorithms more efficiently and comfortably. Algorithm animation also can provide convenient tools to researchers, who are analyzing some existing algorithms or may be searching for a new algorithm based on an existing one. Therefore, in the past few years we have seen the field of algorithm animation flourish.

Geometric animation is an exciting branch of algorithm animation. Many problems in computational geometry are related to points, lines, polygons etc, which can be represented accurately and conveniently by images on computer screens. Geometric algorithms involve 2D and even 3D space problems, which occasionally are harder to be imagined by students or researchers. Geometric animation uses the advantage of computer software and hardware development, such as professional 3D object processing software packages, web-based multimedia authoring platforms and Large Vector General Refresh CRT, to provide a great tool for computational geometry and geometric algorithm research.

During the last several years, more and more multimedia animation experience has occurred on the World Wide Web. Many multimedia tools are programmed within

the constraints of HTML, and then stretched by the enhanced capabilities provided by special plug-ins and players to enable browsers to exceed their limits. The combination of multimedia technology and the Internet technology greatly benefits the distant study and learning of geometric algorithms through the Internet [14].

1.1 PROBLEM STATEMENT

Geometric algorithm animation is a relatively new field in computer science. In this thesis, the recent research progress in this area is introduced. The Internet is quickly evolving from a static, text-based medium to an interactive broadcasting network complete with multimedia content. These multimedia tools make algorithm animation easier to implement on the Internet. A survey of multimedia developing tools and plug-ins tools is provided.

Convex Hull is one of the problems in computational geometry. There are several algorithms to solving this problem; package-wrapping and Graham scan are among them. Convex Hull is a 2D problem that also can be extended to 3D. In this thesis, two 2D Convex Hull algorithms are analyzed, coded, and animated. Then an interactive Shockwave movie of Convex Hull algorithms is generated and can be presented on the WWW.

Geometric algorithms, computer animation, multimedia, and World Wide Web are different areas in computer science. In this thesis, the author tries to cover these four areas and have them work concordantly.

1.2 ORGANIZATION OF THIS THESIS

This thesis is composed of seven chapters. The current chapter explains the motivation and problem of the research and the thesis structure.

Chapter II gives an overview of the development of computer animation, algorithm animation, and geometric animation. The recent progress in these fields is provided.

Chapter III surveys several types of multimedia and WWW-based software that may be suitable for algorithm animation on the Internet.

Chapter IV gives a brief introduction to Macromedia Director, a multimedia software authoring tool created by Macromedia. Several interesting features are introduced, which are employed for programming of Convex Hull Animation Systems like Event Handlers, Messages, and Shockwave movies.

Chapter V illustrates two Convex Hull algorithms: package-wrapping and Graham scan. Time and space complexities of these two algorithms are also given.

Chapter VI presents a detailed design and implementation approach to the Convex Hull Animation System. An important solution for overcoming the shortcomings of Macromedia Director 6 is also presented.

Chapter VII summarizes the thesis and a possible further animation extension is discussed.

CHAPTER II

COMPUTER ANIMATION AND ALGORITHM ANIMATION

This chapter briefly introduces computer animation, algorithm animation, and geometric animation. The recent progress and software packages in algorithm and geometric animation are surveyed. Previous related work finished at the OSU computer science department is discussed.

2.1 COMPUTER ANIMATION

Traditional animation is defined as a technique in which the illusion of movement is created by photographing a series of individual drawings on successive frames of film. A definition of computer animation could be a technique in which the illusion of movement is created by displaying on a screen, or recording on a recording device, a series of individual states of a dynamic scene [31]. Computer Animation is growing very rapidly. However, the theory of computer animation was rather poor until 1989, when many excellent papers were published [30].

Most researchers have distinguished among three computer animation methods: image-based keyframe animation, parametric keyframe, and procedural animation [31]. Keyframe animation consists of the automatic generation of intermediate frames called in-betweens.

- Image-based keyframe animation is an old technique introduced by Burnyk and Wein

- (1971) [9]. The in-betweens are generated by interpolating the keyframe images themselves. The simplest interpolation algorithm is the linear interpolation algorithm.
- Parametric keyframe animation is based on the principle that an entity (object, camera, light) is characterized by parameters. The in-betweens are constructed from the interpolated parameters. A nice algorithm, Kochanek-Bartels spline interpolation, was introduced in 1984 [15].
 - Procedural animation is defined as the motion that is described by a list of transformations, such as rotations, translation etc. The transformation is defined by parameters, which may change in the animation following some physical law. Two programming implementations were given by Reynolds's ASAS (1982) [21] and Thalmann's MIRA (1983) [29].

2.2 ALGORITHM ANIMATION

Algorithm animation started from mathematical visualization and educational film. Scientific data visualization has greatly influenced the recent developments of algorithm animation. The first educational film of algorithm animation was introduced in the late 1970's [3]. In the middle of the 1980's, an algorithm animation system appeared, which allowed the user to control the animation interactively [17]. Recent developments have added multimedia features, like color and sound [6] [10]. The use of object-oriented programming methodology was also considered [7] [25] in the early 1990's. Price, Baecker and Small (1993) [20] gave an excellent overview on the systems used to create general algorithm animation. The field of geometric algorithm animation has received

more interest from researchers because of the difficulties and chances [25][28]. Other excellent work in geometric algorithm animation has been done by Hausner (1996) [14].

2.2.1 GENERAL ANIMATION SYSTEMS

- “Sorting Out Sorting”

The most well-known early algorithm animation is the film “Sorting out Sorting”, created by Ronald Baecker [3] and demonstrated at Siggraph '81. It explains concepts involved in sorting an array of numbers, illustrating comparisons, and swaps [14].

- BALSAs and Zeus

BALSA and BALSA-II [4][5] are general-purpose animation systems, which means that in principle they can be used to animate any algorithm. BALSA and BALSA-II were considered the first major interactive software visualization systems. This early system BALSA was used widely as an aid to teaching algorithm design and analysis and allowed students to view animations prepared by a lecturer interactively.

After completing his work on BALSA, Marc Brown and others developed Zeus [7], an object-oriented algorithm animation system similar to BALSA. Functions such as stop, play, pause, speed control and direction control were provided. Zeus offered a number of enhancements over these earlier systems, in particular the use of color, sound, and more interestingly, 3D graphics [8].

Within Zeus, constructing animations relies on the animator identifying critical areas within the algorithm code, such as data manipulation, conditional evaluation or

procedure calls. These areas are then instrumented with calls to the Zeus "interesting event" manager. During execution of the algorithm, these interesting events will be reported to Zeus, which will then forward them on to many interesting views. The views are predefined and designed to respond corresponding events in the code. One of the interesting features is that during the visualization the user is still able to alter various aspects of a view by changing its associated control panel in a multi-view editor. This allows both generic parameters, like lighting level, and view-specific parameters to be changed easily.

- AnimA and GeoLab

AnimA, a descendant of BALSAs, was developed by Rezende [2] in the early 1990's. AnimA is a general-purpose animation system, which is implemented based on GeoLab [22], a system for geometric computation. AnimA implements many of the features of BALSAs, such as renderers, adapters, and input generators.

GeoLab is very flexible for the user. To run animation, a user can choose a problem, an algorithm for that problem, and an input generator for the algorithm. The user can control the execution of animation by using step through, pause, or abort functions. Dynamic linking allows all the components to be selected at run-time.

2.2.2 GEOMETRIC ANIMATION SYSTEMS

- XYZ Geobench and Workbench

The XYZ Geobench (eXperimental geometrY Zurich) is a workbench for

geometric computation [25]. It provides an interactive user interface similar to a drawing program, which can be used to create and manipulate geometric objects such as points, segments, polygons, etc. Furthermore the user interface provides convenient access with algorithm animation to the XYZ Program Library that contains many standard algorithms for 2D problems and restricted 3D problems.

Epstein, Kavanagh and others [22] published a paper describing their Workbench for computational geometry. They concerned that many published geometric algorithms had not been implemented, and they tried to measure the cost behavior of the implemented algorithms [14].

- Geomview

Geomview, which was created by the Geometry Center at the University of Minnesota (1995) [11], is a sophisticated viewer for geometric data. A user can interactively change the appearance of objects, such as the lighting and surface material properties. Phong shading leads to realistic-looking objects. Moreover, 3D and 4D objects can be viewed with perspective, orthogonal, hyperbolic and spherical projections.

One of its important features is that the appearance and display of geometric data can also be controlled through software. This makes it possible to animate geometric algorithms with Geomview. A programmer needs to modify a program to output descriptions of geometric objects in a Geomview format, and in this way use Geomview as the means to visualize the program's behavior by interpreting the output description. This approach is also the way to animate an algorithm, since Geomview has no built-in

methods for animation.

2.3 PREVIOUS RELATED WORK AT OSU

Much visualization and animation research has been done at the OSU computer science department. In the late 1970's, Adams finished the visualization for two-dimensional linear programming [1]. Some hardware simulation and visualization were also given, such as snoopy cache (1996) [32] and PDP11/40 (1975)[19]. In artificial intelligence, Öz developed a visualization of learning in neural networks in 1997 [35].

Research has also reached to the area of data structure and algorithms. Some animations of the construction of a B-tree have been created using Macromedia Director for course work purposes. Also, a C++ version of B-tree animation has been done by Lin [16]. Xu gave the animation for several basic data structures, including stacks, queues, and lists [34]. Excellent multimedia animation for binary search trees has been done by Shen (1997) [26] using Macromedia Director 6. A rule-based data structure animation was implemented by Harvick in 1997 [13].

CHAPTER III

MULTIMEDIA AND WWW TOOLS OVERVIEW

Multimedia is any combination of text, graphic art, sound, animation, and video delivered by computer or other electronic means [33]. It enhances understanding by involving more of the senses. The World Wide Web started in 1989 as a distributed collaborative hypermedia information system. Now millions of multimedia web pages are posted on the WWW; they are programmed within the constraints of HTML. Since multimedia features based on the WWW are involved, HTML is briefly introduced; plug-ins and several multimedia tools are presented in this chapter.

3.1 HTML

HTML, Hypertext Markup Language, is one of the simplifying components of Web-based applications [27]. HTML documents are simple ASCII text files saved to disk without any formatting at all, so professional Web page developers often use only a word processor and not a dedicated HTML editor, and they insert text and tags into their documents manually or with personalized shortcut keys and helper applications. HTML currently includes about 50 tags, and it is very easy to understand their properties and usage.

3.2 PLUG-INS

Plug-ins add the power of multimedia to Web browsers by allowing users to view and interact with new types of documents and images [27]. Helper applications or players also provide multimedia power by displaying or running files downloaded from the Internet by browser, but the helper/players starts up and runs on the user's computer separately from the browser. Plug-ins can not run independently without a browser.

Many plug-ins are designed to perform special tasks not available without the plug-in installed. To get a plug-in, the user accesses the Web site of the vendor of the plug-in and downloads the version suited to the user's platform and browser. Then, when the user's browser comes across a file of a MIME-type supported by the user's plug-in, it transfers operations to the plug-in, doing what that plug-in was designed to do. Table 3-1 shows some plug-ins supported by Netscape Navigator:

Table 3-1: Multimedia Plug-ins and Vendors

Name	Company	URL
QuickTime VR	Apple	http://www.apple.com
Live3D	Netscape	http://www.netscape.com
RealAudio	Progressive Networks	http://www.realaudio.com
WebAnimator	Site Technologies	http://www.sitetech.com
PreVU	InterVU	http://www.intervu.com
QuickSilver	Micrografx	http://www.micrografx.com
Shockwave	Macromedia	http://www.macromedia.com

3.3 MULTIMEDIA AUTHORIZING TOOLS

Multimedia authoring tools provide an important framework for a user to organize and edit the elements of a multimedia project, such as graphics, sound, animation, and video clips. Multimedia authoring tools also offer developers an integrated environment for designing GUI and assembling all multimedia elements into a single cohesive project.

Generally speaking, an authoring system provides the following functions,

- Creates, edits, and imports specific types of data.
- Binds raw data into playback sequence or cue sheet.
- Provides a structured method or language for responding to user input.

Vaughan (1996) [33] proposed an idea to group the current existing authoring tools based on the metaphor used for sequencing or organizing multimedia elements and events. They are,

- Card-based or page-based: Multimedia elements are organized as pages of a book or a stack of cards.
- Icon-based and event-driven: Multimedia elements and interaction events are organized as objects in a structural framework or process.
- Time-based: Multimedia elements and interaction events are organized with a timeline, using resolution 1/30 second or lower.
- Object-oriented: Multimedia elements and interaction events are objects living in a hierarchical order of parent and child relationships

3.4 CARD-BASED OR PAGE-BASED TOOLS

A simple and easily understood metaphor for organizing multimedia elements is provided by card-based and page-based systems. Routines, which are created by using system organization tools or scripting language, simply direct the multimedia elements to the page or card containing appropriate images and text along with sound animation and video clips.

An object can be anything in these systems such as button, text field, graphic object, background, page or card, and even the routine script or project. An object has properties used to describe its characteristics; also an object can contain a scripting routine, which is activated when an event related to this object occurs. Systems provide many event-handlers that can be customized by the user for special purposes. Systems monitor event messages. If these messages occur, corresponding event handlers are activated. Several popular card-based and page-based systems are briefly introduced as follows:

- HyperCard: Starting in 1987, it is the most widely available programming system and multimedia-authoring tool for the Macintosh.
- SuperCard: An authoring system based on Macintosh and Windows systems that can produce sophisticated multimedia presentations, front-ends to databases, and computer-based education and training projects [33].
- ToolBook: A Windows authoring tool that can create multimedia projects to present information graphically as drawings, scanned color images, text, sounds, and animations.

3.5 ICON-BASED TOOLS

Icon-based tools are a visual programming approach to organizing and presenting multimedia presentations. The key step of these tools is to design a flowchart at the beginning. Each element in the flowchart is one of the events, tasks, and decisions, which are represented by icons in a library. These icons can include menu choices, graphic images, sound, and computations. The flowchart defines the logic structure of a project. When this logic structure is built, a multimedia element such as text, images, animation, sound and video clips can be added. The flowchart can be modified if the user wants to. Several popular icon-based authoring tools are briefly introduced as follows:

- Authorware Professional: An easy-to-use tool that even non-technical authors can generate sophisticated applications without using any scripting control. It works both on Macintosh as well as Windows platforms. Thirteen logic and multimedia icons are provided in the system: Display icons; Animation icons; Erase icons; Wait icons; Decision icons; Interaction icons; Calculation icons; Map icons; Start Flag icons; Stop Flag icons; Movie icons; Sound icons; Video icons.
- Quest: A product from Allen Commination that uses a workflow organization with arrows delineating branching paths. It provides two levels of development: Design Level and Frame Level. Quest also includes an embedded scripting language, Quest C, borrowing the power from ANSI C.

3.6 TIME-BASED TOOLS

Time-based tools are a time programming approach to organizing and presenting

multimedia presentations. A visual timeline (score) is used to sequence the events of a multimedia presentation. The display of each multimedia presentation is arranged by timeline as precisely as possible. The user can manipulate a frame and its duration inside the timeline. Time-based authoring tools are the most common of multimedia authoring tools. Macromedia Director is one of the most popular time-based authoring tools; it is employed in the implementation of the Convex Hull animation system. In Chapter IV, an introduction for Macromedia Director is provided.

3.7 OBJECT-ORIENTED TOOLS

Object-oriented programming (OOP) methodology is commonly accepted and applied in many aspects in software engineering. In an object-oriented authoring system, every multimedia element is considered an object, such as graph, text, animation, video clip and sound. Properties are used to describe the characteristics of every object. When events occur, objects respond and send messages to each other. An object can inherit the behaviors of another object. No timelines or stacks of cards are used to organize the multimedia objects; instead, a hierarchy of sections, subsections, and scenes are built at the beginning. New objects are dragged from an object palette and dropped into a project space. Following are some object-oriented multimedia authoring tools,

- mTropolis: mTropolis begins a project by building elements in empty space. The hierarchy is used to organize all objects. Like many other multimedia authoring tools, it is also a message or an event driven application system. Messages are broadcast over the whole project and some corresponding objects are activated. When in a

parent-child relationship, an object can respond to messages as a family [33]. MTropolis offers two viewers, Layout View and Structure View, to help the user organize the entire project. Layout View manipulates image aspects and Structure View manages the logical hierarchy of objects in a project.

- **MediaForge:** MediaForge is very flexible for users. The user can work with an object-based authoring system, or just work with scripting language, or both. When using its object-based features, it provides a List Editor to manipulate all of the objects. Hierarchy of objects is represented by a family tree.
- **Apple Media Tool:** Apple Media Tool is an organizer and player of media [33]; no editor and creator are provided in this system. Therefore, any multimedia elements should be created by other dedicated tools before using. The backbone of this authoring tool is *Screens*, which is made up of objects that point to a media type or remain empty simply as controllers of events.

CHAPTER IV

MACROMEDIA DIRECTOR

4.1 MACROMEDIA DIRECTOR OVERVIEW

Macromedia Director is a powerful developing platform from Macromedia with a broad set of features to create interactive multimedia presentations. As mentioned in the previous chapter, it is a time-based authoring tool. In Macromedia Director, one of the exciting features is the interactive Shockwave movie, which meets the growing need for Internet/Intranet multimedia presentations [23].

Macromedia Director makes a user assemble and sequence the multimedia elements, such as graphics, text, sound, animation and video clips, by using the time-based tool *Score* and the media element database tool *Cast*. When a presentation runs, Director works as a coordinating resource, taking media-related data and whisking it in and out of the computer in the sequence the user specifies.

It is possible to concoct a multimedia piece entirely within Macromedia Director using solely these built-in capabilities. But most multimedia elements are generated by a dedicated software package, such as Photoshop for images and SoundEdit for music. Macromedia Director imports them only when they are ready for final assembly.

The major properties of Director are the following:

- Superior animation. Director uses a frame-by-frame metaphor, which makes it easy for Director to control the finer elements of action. With Director 6, the degree of

control is even further refined, with new features that let the user precisely synchronize sound and motion.

- Platform Independent. Director uses Idealized Machine Layer (IML), which maximizes portability by keeping the multimedia data isolated from the system-specific data. IML makes software, generated by Director, compatible to Windows, Macintosh and any other operating system.
- Extensibility. Today it is easy and efficient to upgrade hardware in a computer. However, software has a different story. For example, the neat new features from a software package Version 5.5 cannot be added to Version 5.0. Version 5.0 has to be thrown out to make way for 5.5. But in Macromedia, Xtras and Xobjects are added to extend and improve upon Director's features.

4.2 BASIC CONCEPT

Cast: A multimedia database contains still images, sound files, palettes, text, programming scripts, movie clips and even other Director files [33]. The user can import a large diversity of multimedia elements, generated by other multimedia tools, into *Cast*, as well as create multimedia elements by using Director's built-in tools.

Score: It is a sequencer for displaying, animating, and playing *Cast* members. *Score* contains the notation that describes your movie and is the primary tool for creating and editing animation. The *Score* window contains a record of everything that happens on the Stage. *Score* provides elaborate and complex visual effects and transitions, adjustments of color palettes, and tempo control.

Lingo: *Lingo* is the full-featured scripting language of Macromedia Director. It expands interactivity and programmed control of movie and animation. It allows users precisely control text, sound, image and digital video [24]. Because of *Lingo*, users can create interactive movies that play on the Internet/Intranet; users can explore and travel through in the order that the user thinks is best. It also can combine animation and sound that Score alone can not. Today, there are the following distinct types of *Lingo*:

- *Core Lingo*: the scripting syntax and terminology that's built into Director itself.
- *NetLingo*: language developed to integrate Shockwave movies into the Internet environment.
- *Xtra Lingo*: a simple additional language made comprehensible to Director.

4.3 EVENT AND EVENT HANDLER

An event is an action that will automatically trigger a script known as an event handler. In Macromedia Director 6, thirteen events are captured by system, shown in Table 4-1.

Event handler is a script that is intended for execution when a *Lingo* event occurs. The event, or it may be called a message, is broadcast when it occurs. The corresponding event handler is activated when an event occurs [23]. In Director 6, the user can customize these event handlers by adding *Lingo* script. Director 6 offers four types of *Lingo* script, which can be added into corresponding event handlers for different purposes.

- *Score* script: The script is assigned to sprites or frames in the *Score*. The hierarchy of

this script is shown in Figure 4-1.

- *Cast member script*: The script is attached to a case member, which is independent of the *Score*. Shown in Figure 4-2.

Table 4-1: Events provided by Macromedia Director 6

Event	Description
Mouse Up	A mouse button was released.
Mouse Down	A mouse button was clicked.
Right Mouse Up	The right mouse button was released. (On the Macintosh, Director treats Control-clicking the same as clicking the right mouse button on Windows system's keyboard.)
Right Mouse Down	The right mouse button was clicked.
Mouse Enter	The pointer entered a sprite's region.
Mouse Leave	The cursor left a sprite's region.
Mouse Within	The cursor is within the sprite's region.
key Up	A key was released.
key Down	A key was pressed.
Prepare Frame	The playback head has left the previous frame, but has not yet entered the next frame.
Enter Frame	The playback head entered the current frame.
Exit Frame	The playback head exited the current frame.
New Event	A specified message was received from a script or behavior.

- **Movie script:** The script is available during the entire movie. It can not attach to a specific object. Shown in Figure 4-3.
- **Parent script:** A special type of script that contains Lingo used to create child objects.

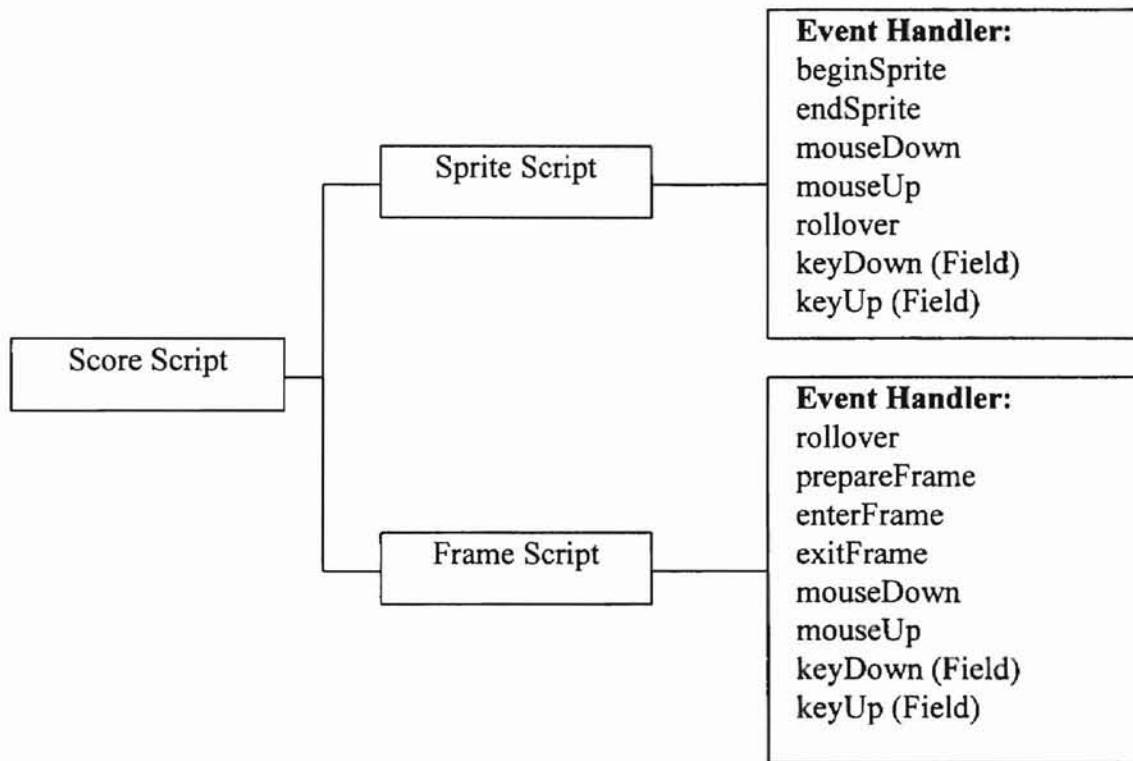


Figure 4-1: Score Script Hierarchy and Handlers

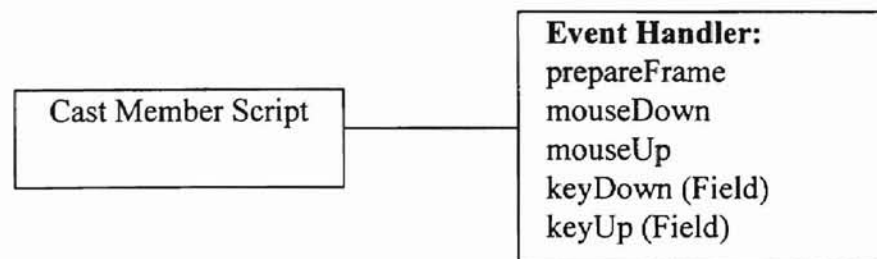


Figure 4-2: Cast Member Script and Its Handlers

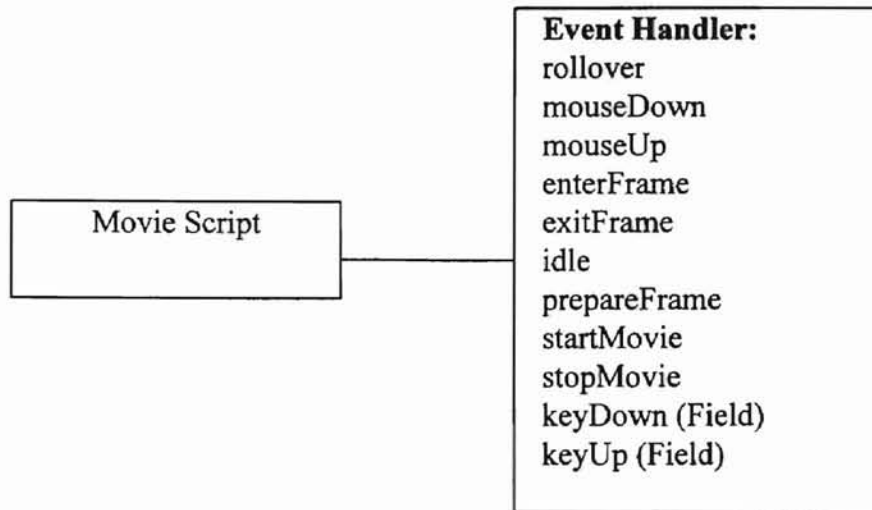


Figure 4-3: Movie Script and Its Handlers

4.4 SHOCKWAVE MOVIE

One of the most exciting features in Macromedia Director is Shockwave movie, a Director Net result, which can be presented on the Internet or Intranet. IML (Idealized Machine Layer) makes Shockwave technology possible. This technology is used to compress, encapsulate, and embed Director movies in HTML documents used on the WWW. It is also used to refer to the plug-in necessary to view such movies in online browsers such as Netscape Navigator or Internet Explorer. "Since Shockwave is essentially another porting process for Director, it is quicker to turn multimedia into Internet content than its competitor Java script today" [23].

CHAPTER V

CONVEX HULL ALGORITHMS

5.1 CONVEX HULL ALGORITHMS

Today geometric problems are receiving more and more attention because of the increasingly widespread use of graphics and the advent of CAD/CAM [18]. Convex Hull is one of the geometric problems. The introduction to two Convex Hull algorithms, *package- wrapping* and *Graham scan* [12], is given.

Convex and Convex Hull Problems are given as follows:

- Convex: A set of points in space is said to be convex if and only if, given any two points, P_1 and P_2 , in the set, all points on the line segment $\langle P_1 P_2 \rangle$ are in the set as well.
- Convex Hull problem: Given a finite set of points in n -dimensional Euclidean space, find the summits of its convex hull [12], as shown in Figure 5-1.

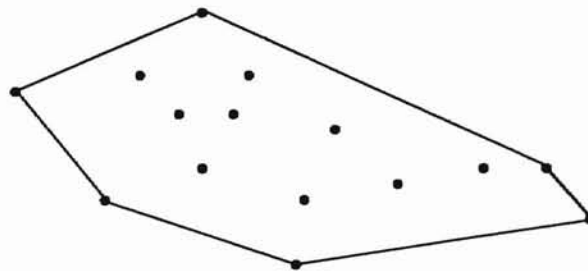


Figure 5-1: The Convex Hull of A Set of Points

5.2 PACKAGE-WRAPPING ALGORITHM AND ANALYSIS

The package-wrapping algorithm is a simple algorithm based on a physical interpretation of the problem. It is similar to a person solving the problem intuitively: “wrap up” the point set, as shown in Figure 5-2. The following is the pseudocode of the algorithm:

Find a point P_1 with minimum Y coordinate. If more than one point, breaking the ties by using the minimum X coordinate.

(This point is the “anchor” point, guaranteed to be on the convex hull.)

Take the half-line from P_1 in the positive direction of the x -axis and sweep it up until it hits another point P_2 ; this point must be on the convex hull.

(If two or more points are hit at same time, break the ties by using their distances from P_1 .)

Execute the following step until reach point P_1 .

For each value of $k=2, 3, \dots, N$, find the point P_{k+1} by sweeping the half-line $\langle P_{k-1} P_k \rangle$, breaking the ties by using their distances from P_k .

The time complexity of the package-wrapping algorithm is $\Theta(nm)$, where n is the number of points in the set and m is the number of points on the convex hull. Because

each step changes the perspective from which the vertices are seen, the angles have to be recomputed at each step. This is the main cause of inefficiency in the package-wrapping algorithm. In the worst case, all the given points lie on the hull and the package-wrapping algorithm takes quadratic time. The Graham scan algorithm uses a fixed perspective to improve the time complexity from the package-wrapping algorithm.

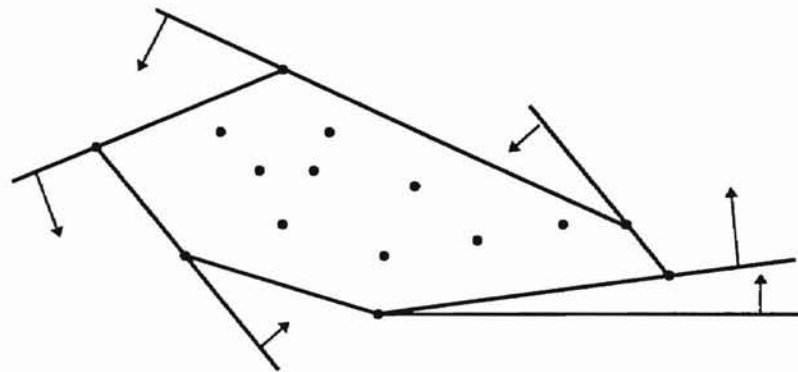


Figure 5-2: An Example of Package-Wrapping

5.3 GRAHAM SCAN ALGORITHM AND ANALYSIS

The following property makes the Graham scan much faster than the package-wrapping algorithm; that is, if a point is known to lie inside the convex hull, it need never be reconsidered. An example of the algorithm is shown in Figure 5-3. The algorithm can be summarized as follows:

Selecting perspective: Find a point P_1 with minimum Y coordinate. If more than one point, break the ties by using the minimum X coordinate.

(This point is the “anchor” point, guaranteed to be on the convex hull.)

Sorting: Using P_1 as the origin, sort the remaining points in ascending order by the angles formed by the positive direction of the x -axis and by

the lines from P_1 to each point, breaking ties by using their distances from P_1 . Give the subscribe of points corresponding to the sorted angles: P_1, P_2, \dots, P_n .

Graham scan:

Push P_1, P_2 into a stack. The stack holds a partial convex hull.

(The points inside the stack are represented by $S_1, S_2, \dots,$

S_k .

S_k is the point on the stack top.)

Add the next point P_j to the partial convex hull ($j=3,4,\dots,n$).

Eliminating points that are not on the convex hull from the stack:

Draw segment from S_{k-1} to P_j

If S_k is left of this segment

Pop S_k (Eliminate S_k)

Repeat while elimination continues

Push P_j

Closure:

Connect S_k to P_1 to complete the convex hull.

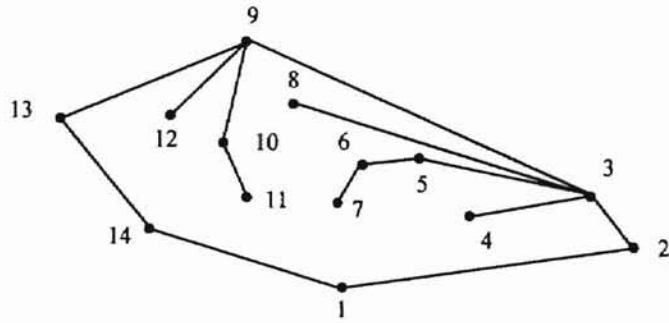


Figure 5-3: An Example of Graham Scan

With respect to a fixed perspective, the points can be ordered in $\Theta(n \log n)$ time by the preprocessing step. Each point is pushed onto the stack exactly once and removed from the stack at most once; thus, the scan phase takes $\Theta(n)$. Therefore, the time complexity is $\Theta(n \log n)$. The Graham scan algorithm is faster than the package-wrapping algorithm, especially when all of the points are on the convex hull.

CHAPTER VI

SYSTEM DESIGN AND IMPLEMENTATION

Macromedia Director provides many powerful functions that greatly benefit the implementation of multimedia geometric animation systems. Since Macromedia Director mostly emphasizes multimedia element control rather than geometric object manipulation, several problems need to be solved and shortages need to be made up in animation system design and implementation. At the beginning of this chapter, a flowchart gives an overview of the entire system design. An analysis of the entire system is also given. Several problems that occur during the period of system design and implementation are discussed and appropriate solutions are proposed. Parent/Child script, puppet sprite and their applications and advantages are presented. Finally the application of Shockwave movie briefly is introduced.

6.1 SYSTEM ARCHITECTURE

The system consists of seven modules; most of these modules are coded by sprite script and movie script. The *Algorithm* module is coded using Parent/Child script, which gives birth in the *Control Panel* module during the runtime when the *Algorithm* module is needed. This saves memory space for the entire project. The architecture of the seven modules and the relationship among them are shown in Figure 6-1.

The core of the animation system is the module *Control Panel* and the module *Animation Control*. A developer can add any other algorithms by rewriting the *Algorithm*

module; then different animation will be generated. The only restriction on the algorithms added by the developer is that they should be based on point and line operations.

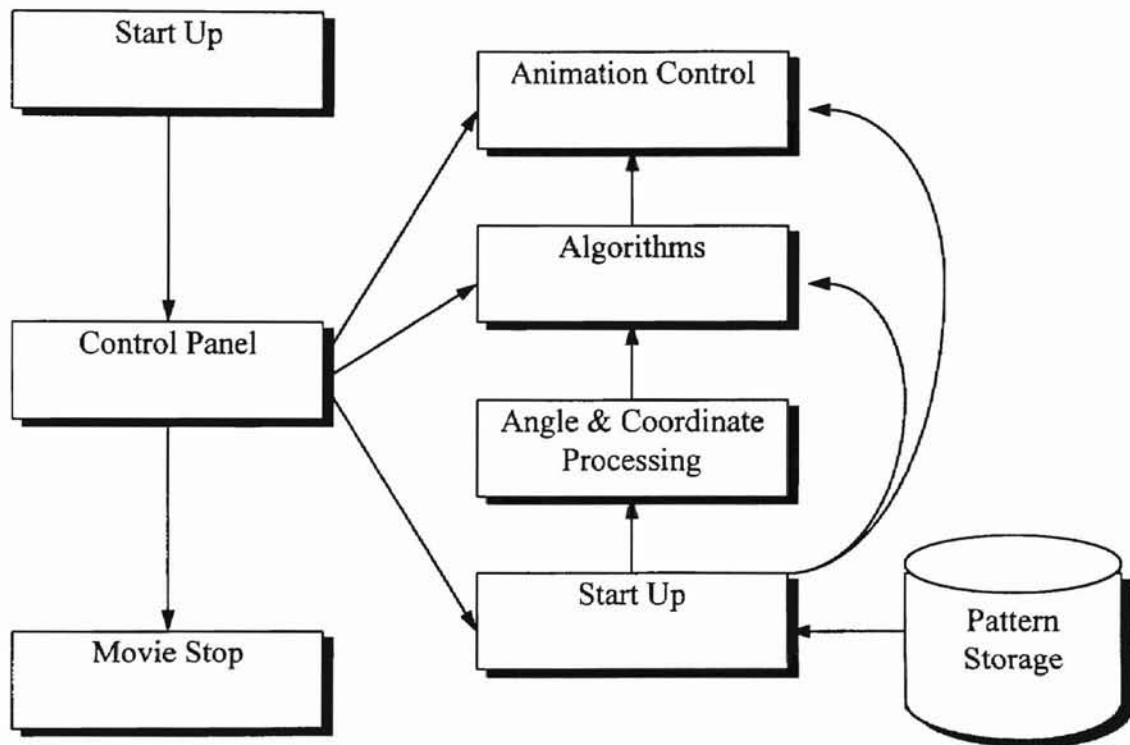


Figure 6-1: The System Architecture

First of all, the module *Start Up* initializes all of the global variables and global lists and set the playback head to frame 1, which makes the Stage show on the center of the screen. Then the module *Control Panel* takes over control and shows several pushdown buttons that allow the user to make selections and manipulate the entire animation process.

The module *Animation Control* processes all requests from any other modules, such as drawing a line and drawing a point or deleting a line and deleting a point. The module *Algorithms* provides parameters about the position and shape of geometric

objects that the module *Animation Control* needs to display or erase on the Stage. Because of Macromedia Director's limitation, angle calculation and coordinate transformation are necessary, which is the task of module *Angle & Coordinate Processing*.

The module *Input* not only allows the user to input points by clicking the mouse over the Stage, but also provides several pre-provided patterns to demonstrate the difference between two algorithms; therefore, a pattern storage is provided for this purpose.

6.2 LINE AND POINT MANIPULATION

It is essential in geometric animation, to manipulate points, segments, lines, and other geometric objects like polygons, and circles in 2D space and cubics, spheres in 3D space. The shape and position of geometric objects may be changed along with the states of geometric algorithm execution. Thus, full control of the shape and position in geometric animation is important. Macromedia Director Studio provides many methods to control the position of any multimedia object on the Stage that is a pre-drawn geometric object. Unfortunately, the shape of a pre-drawn geometric object can not be changed by modifying its shape properties, and there is no built-in function provided to draw an object. Therefore the shape control in Macromedia Director is very limited.

Through the study of Macromedia Director, an idea comes out. Director provides sprite script to manipulate a sprite. A user can use sprite script to implement the movement and stretch by changing some properties of the sprite. Sprite is an individual instance of a cast member on the Stage, as documented by the placement of that cast

member in the Score. Sprites can display several qualities not shared by the source cast member, but changes to the cast member are immediately reflected in all sprites derived from that cast member. Because of the restriction of Score, the sprite script of a sprite can control this sprite in just one frame time, but geometric animation needs to control a sprite across the entire animation period that may contain many frame times. Director provides a puppet sprite to extend the capabilities of manipulating a single sprite. When a Director element such as a sprite is puppeted, this sprite is controlled by Lingo script rather than by instructions embedded in the Score. More precisely speaking, puppeting is a condition applied to the channel the sprite occupies rather than the sprite itself. For example, a sprite may be derived from one cast member, a vertical line, but when puppeted, another cast member, a horizon line, may be substituted. Using the puppet sprite and sprite script extend the capability to draw and control an arbitrary line on the Stage, although many technical problems still need to be solved. A detailed implementation approach and some solutions for specific problems are presented as follows.

- Flow Chart of Draw Line Function

A flow chart of the draw line function is shown in Figure 6-2. At the beginning, $(X1, Y1)$ and $(X2, Y2)$ are transferred into this function, which is used to describe a line (segment) on the Stage. To initialize Line-List, a list stores the sprite line number and usage status; the program sets all of puppet sprite line *unused*. When drawing a line, the angle between the line and y -axis' positive orientation is calculated; the program uses the

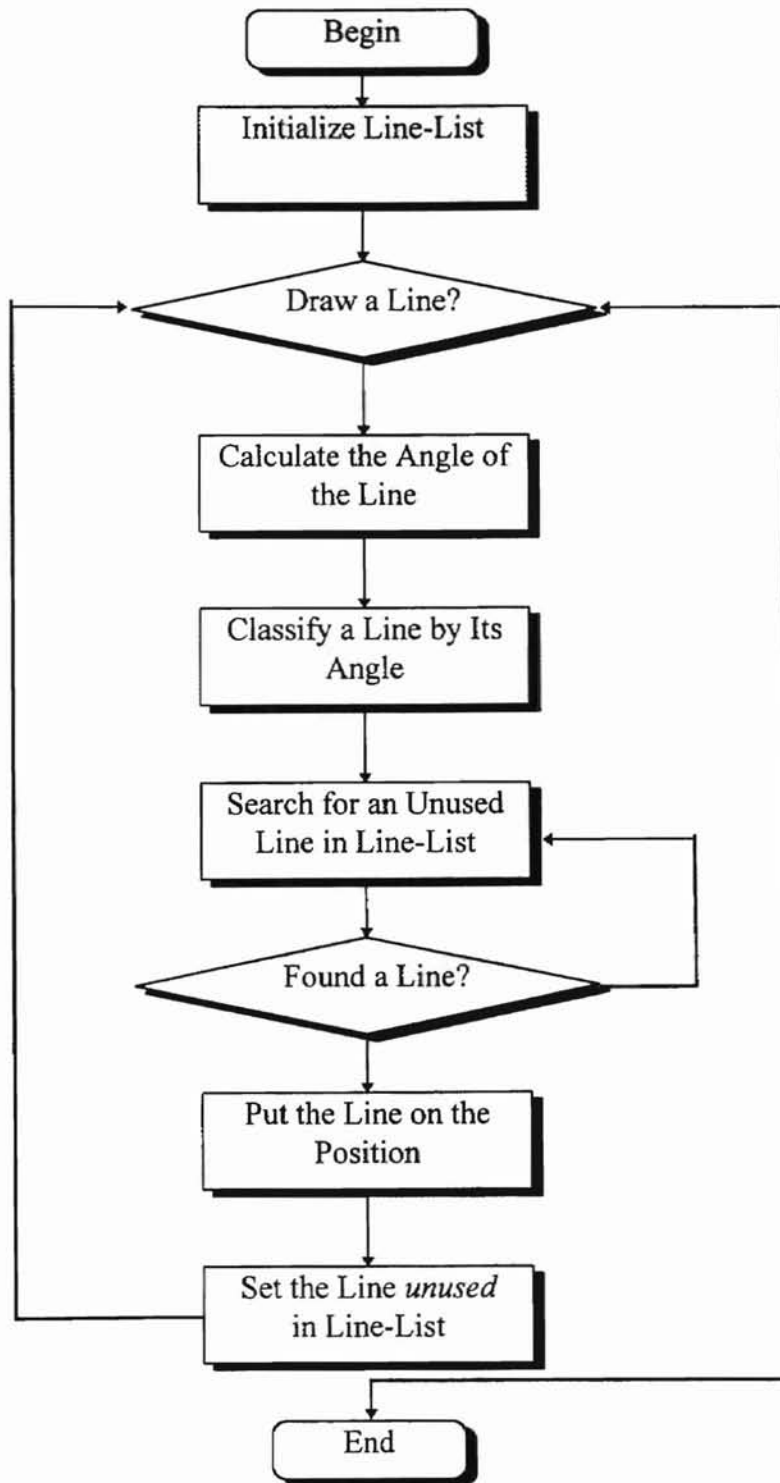


Figure 6-2: Flow Chart of Draw Line Function

angle to classify the line into two categories: the line in the first and the third quadrant (*line1*) and the line in the second and the fourth quadrant (*line2*). The reason for this classification is explained later. The Line-List contains equal numbers of *line1* and *line2*. After classifying a line into one of the *line1* and *line2* categories, the program tries to find an unused line in the Line-List. If found, it puts the puppet sprite line at the position described by (X1, Y1) and (X2, Y2).

- Line Format and Its Performance

A line must be created by a dedicated image tool and put into the Cast as a cast member before it is used on the Stage. Macromedia Director offers two built-in tools, *paint tool* and *tool palette*, to create graphic objects. The *paint tool* generates a **bit-mapped** format graphic. This means all elements of its artwork are stored and displayed as arrangements of pixels. When a line is created by *paint tool*, it is stored in the bitmapped graphic format. Instead of this, *tool palette* draws a line using a Macromedia graphic format, which takes up less file space than bitmapped graphics and they are easily modifiable by changing some settable properties at any point after their creation. The most important advantage of *tool palette*'s line is that it can be stretched in a big range of angle and length changes. In contrast with *tool palette*'s line, *paint tool*'s line is less stretchable and always becomes a dashed line when the angle and length change a lot from the original image. When approaching some special angles like 0 or 90 degrees, *paint tool*'s line even disappears. The comparison of the tool palette line and paint tool line in different ranges of angle change is shown in Figure 6-3, where T is the angle

between a line and y -axis positive direction. Figure 6-3 gives five different angles: $T= 30$, 15, 10, 5, and 0.2 degrees. The performance of a *tool palette* line is much better than that of a *paint tool* line. Another advantage of *tool palette*'s line is that the color and pixel properties of a line can be set in Lingo script; that is, a user can control the color and thickness of a sprite line in animation. When using a *paint tool* line, the user can just get the color property but the color and pixel properties can not be set. A *tool palette* line shows more advantages in line manipulation and demonstration; as a result, a *tool palette*'s line is employed in this animation system.

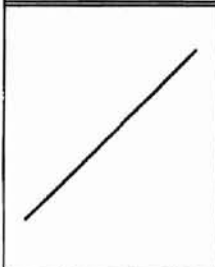
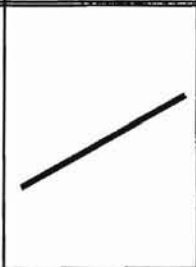




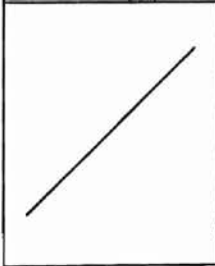


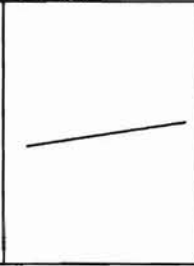
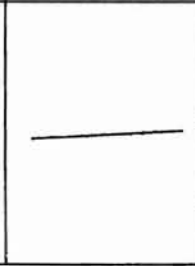

<i>Paint Tool Line</i>					
Cast Member	Sprite (T=30)	Sprite (T=15)	Sprite (T=10)	Sprite (T=5)	Sprite (T=1)
					
<i>Tool Palette Line</i>					
Cast Member	Sprite (T=30)	Sprite (T=15)	Sprite (T=10)	Sprite (T=5)	Sprite (T=1)
					

Figure 6-3: Comparison of Line Implementation by *Paint Tool* and *Tool Palette*

- How many cast member lines is enough for presenting all sprite lines?

Macromedia Director provides the function *rect* to set both a sprite dimensions and position. The syntax is,

rect (left, top, right, bottom) or *rect* (point1, point2)

This function has two formats: When using four arguments, the *rect* function defines a rectangle that has the sides specified by left, top, right, and bottom. The left and right values specify the number of pixels from the left edge of the Stage. The top and bottom values specify the number of pixels from the top of the Stage. When using two arguments, the *rect* function defines a rectangle that encloses the points specified by point1 and point2.







Cast member line lies on the same quadrants as puppet sprite line lies on		
The cast member line	Point1, Point2 on the Stage	Puppet sprite line connects two points, very well!
		
Cast member line lies on different quadrants from the puppet sprite line		
The cast member line	Point1, Point2 on the Stage	Puppet sprite line can not connect two points, bad!
		

Figure 6-4: Comparison of Lines in Different Quadrants

When the demonstration of a line on the Stage is needed, the user draws a line by using *tool palette* and puts it into the *Cast* as a cast member. Then the user copies this cast member line into *Score* and sets it as a puppet sprite line in order to fully control it by using Lingo script. After this, the user sets the *rect* (point1, point2) of this puppet sprite line, where point1 and point2 are end points of the line wanted to show on the Stage. If the cast member line lies on the same quadrants as the line wanted to show on the Stage, the puppet sprite line on the Stage will fit precisely between two points (point1, point2). But if the cast member line lies on different quadrants from the line wanted to show on the Stage, the puppet sprite line on the Stage will rotate 180 degrees from the line the user needs, as shown in Figure 6-4. To solve this problem, we use two cast member lines, with degree 45 and 135 respectively shown in Figure 6-5, instead of only one line to represent almost every line on the Stage.

The above approach is excellent for most lines except the lines with angles 0, 90 (180, 360), since these two lines can not even show on the Stage. One idea is to change a vertical or horizontal line slightly and make it not vertical or not horizontal by adjusting one of the two end points of the line. For example, a vertical line with two end points

Point1: X1=100 and Y1=300

Point2: X2=293 and Y2=300,

we can adjust the Y2 of point2 by add 1 pixel, that is, drawing a line between Point1 (100, 300) and Point2 (293, 301). Actually, this non-vertical line has almost the same visual effect as a vertical line. Another idea is to use four lines, one vertical and one horizontal, along with two lines shown in Figure 6-5, to implement draw line function. But this approach complicates the management of sprite lines. Therefore, in this thesis the

implementation of drawing an arbitrary line on the Stage is by using two cast member lines, as shown in Figure 6-5.

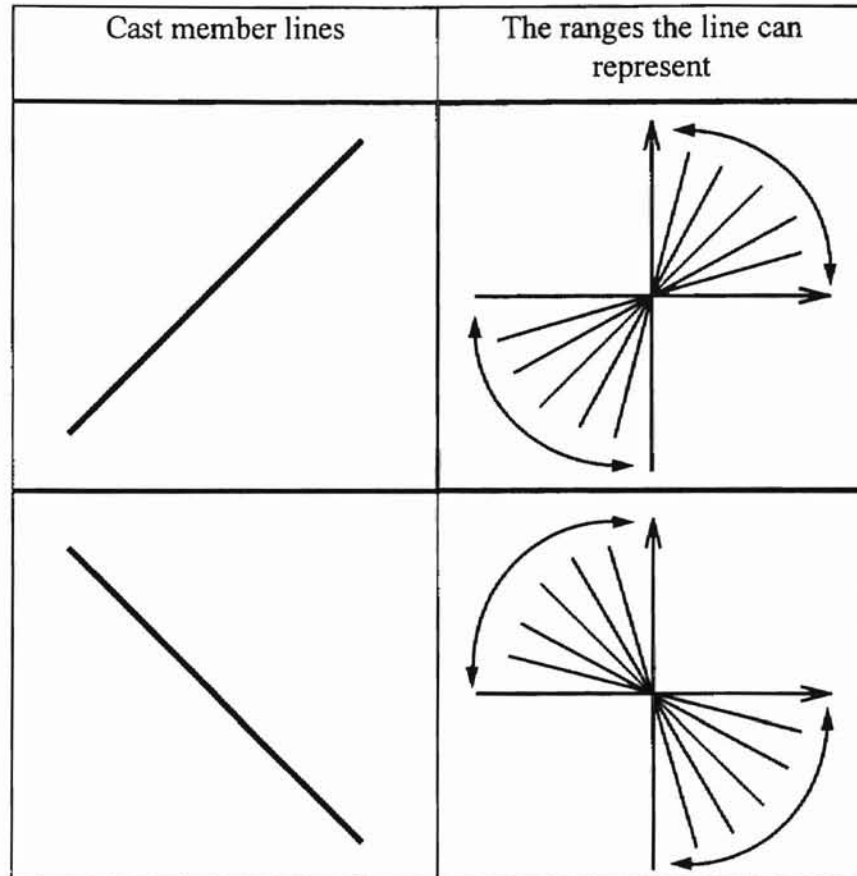


Figure 6-5: The Cast Member Lines and the Ranges They Represented

- Puppet sprite line management

When a line is shown on the Stage, this line should be recorded into a list called Line-List by setting the *used* bit to 1. Because of the limitation of the *Score* - the total number of channels is 120 - every channel just allows one puppet sprite to exist. Thus, we have less than 120 puppet sprites that can be used to draw lines on the Stage. If more than

120 lines are needed and some lines are deleted during animation, we should reuse these deleted lines by setting their *used* bit to 0. When the next drawing line request occurs, the program searches for an unused puppet sprite in the Line-List by determining its *used* bit as 0. This solution makes it possible for users to draw more than 120 lines during animation. It greatly extends the capability of the draw line function.

- Draw points

Drawing points is similar to the draw line function. It uses the same management approach to recording a drawn point by setting the *used* bit to 1 in the Point-List. When a point is deleted, the *used* bit is set to 0 and it is ready for reusing. Unlike drawing lines, a point just needs an X and Y to describe its position on the Stage; no stretch and dimensions are needed. Therefore the draw point function is much simpler than the draw line function.

6.3 ANGLE CALCULATION AND PROCESSING

In the Convex Hull algorithm, most steps require accurate and efficient determination of the value of angles. But the calculation of an angle value should use some transcendental functions, which are time consuming. Through analysis [18], all of these angle values are used just to compare themselves. As we need only compare angles, however, we can avoid computing their actual values and thus avoid using transcendental functions.

A function in direct proportion to the angle is needed, that is, a function, f , such that $f(x) < f(y)$ whenever $x < y$, where x and y represent angles in the range $[0, 2\pi)$

Several trigonometric functions, such as the sine, have the desired property in the first quadrant and can be computed from the Cartesian coordinates without first determining each angle. We need a function with the desired property in all of four quadrants. A function called a pseudoangle function has been introduced by Moret and Shapiro [18].

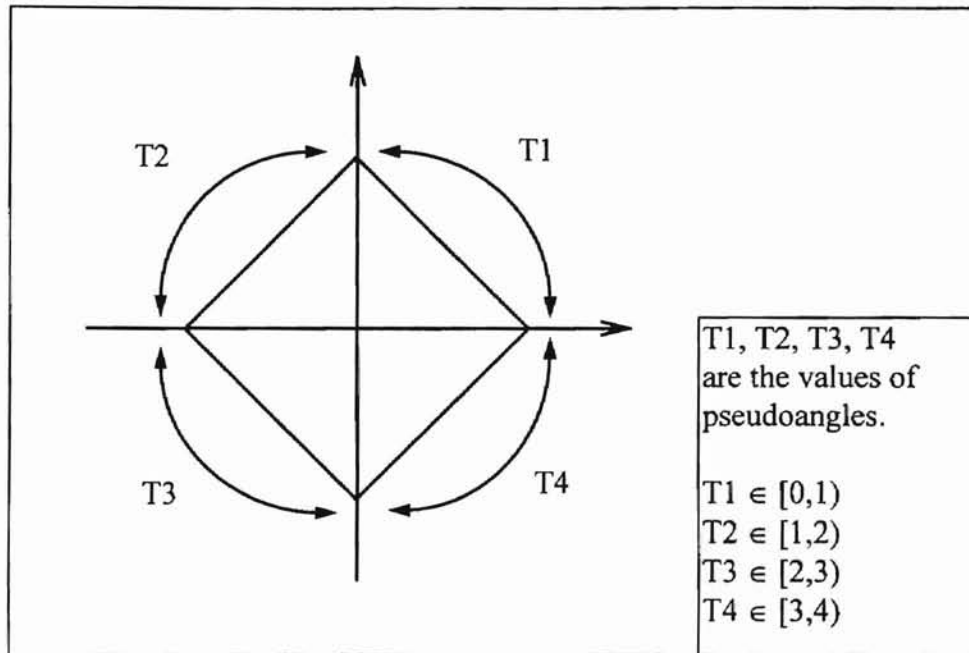


Figure 6-6: Pseudoangle and Its Range

This function is called a pseudoangle function because it behaves like an angle under comparison and because its range closely approximates that of a real angle: it is $[0,4)$ instead of $[0,2\pi)$. Based on a comparison of angles, this function is better because it uses only one division and no multiplication or square root. This function is much like the sine function in the first quadrant, varying from 0 to 1 as the angle varies from 0 to $\pi/2$. But the hypotenuse length is computed according to the Manhattan metric rather than the

Euclidean metric [18], with the result that the pseudoangle replaces the unit circle, on which the sine is based, with the triangle formed by the axes and the line connecting (0,1) to (1,0). The pseudoangle value and its corresponding quadrant are described in Figure 6-6. Calculation of the pseudoangle is given in Figure 6-7.

```
On Theta (X1, Y1, X2, Y2)
  set dX = float (X2 - X1)
  set dY = float (Y2 - Y1)

  if dX=0 and dY =0 then
    set angle = -1.0
  else
    set t = dY / ( abs (dX) + abs (dY) )
    if dX < 0.0 then
      set t = 2.0 - t
    else if dY < 0.0 then
      set t = 4.0 + t
    else
      set angle = t
  end if
  return angle
end Theta
```

Figure 6-7: The Pseudoangle Calculation

Although we have a function to calculate pseudoangles to allow for arbitrary shifts of origin, the reference axis must remain parallel to the x -axis and have the same orientation, and the pseudoangle is based just on the reference axis. That is enough for the first scan from the anchor point in a package-wrapping algorithm, but for the following scans in the package-wrapping algorithm and in the Graham scan algorithm, an angle between two arbitrary lines is also needed.

If two lines lie on the same quadrant or two lines lie on two adjacent quadrants, the angle between the two lines can be simply obtained from the absolute value of

the angle of line 1 - the angle of line 2

But this will cause a problem if two lines lie on two non-adjacent quadrants; an example is shown in Figure 6-8. In this example, we want to compare the angle $\angle(P1, P2)$ with the angle $\angle(P1, P3)$. There are

$$\angle(P1, P2) = T1 - T2$$

$$\angle(P1, P3) = T1 - T3,$$

thus, we get the result: $\angle(P1, P2) > \angle(P1, P3)$.

But in Figure 6-8, we can see that $\angle(P1, P2)$ is smaller than $\angle(P1, P3)$. To solve this problem, we use a complementary angle if an angle is larger than π , or larger than 2 in pseudoangle. Therefore, if a pseudoangle is larger than 2, the value of the angle is 4 minutes the pseudoangle.

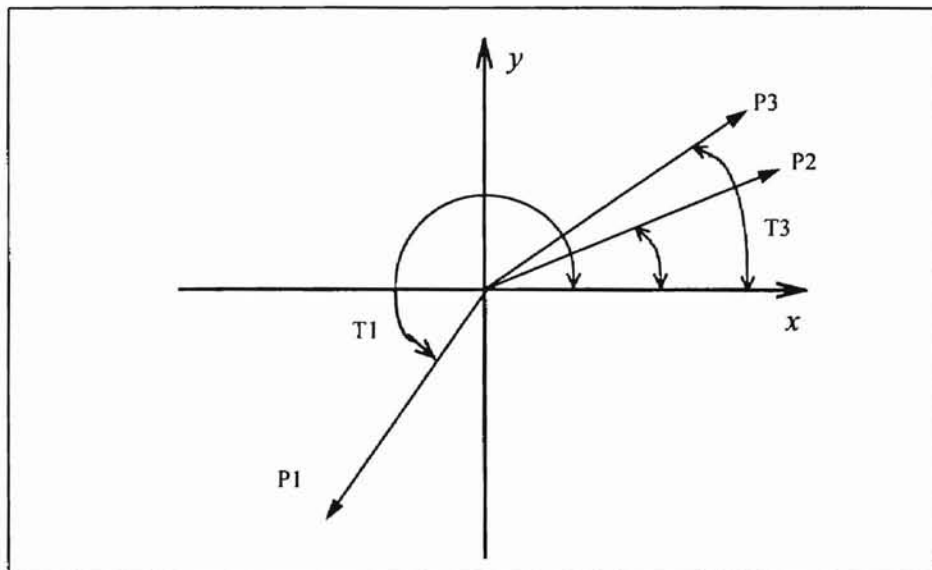


Figure 6-8: An Example of Angle Calculation

6.4 COORDINATE SYSTEM TRANSFORMATION

Stage is the non-moveable window in Director that represents the action transpiring during playback. When a projector is created from a Director movie, the Stage window becomes the only visible window. The Convex Hull animation is presented on the Stage, but the coordinate system on the stage is different from a common coordinate system. Although the algorithm animation will work well on Stage's coordinate system without any changes, it still looks strange when an anchor point is found in the top of a set of points on the Stage. As shown in Figure 6-9, the y -axis is on the opposite orientation from the common y -axis, and the origin is at the left top of the screen. If the height and width of the Stage are H and W respectively, we use the following equation to transfer the Stage coordinates to common coordinates.

$$X_c = X_s$$

$$Y_c = H - Y_s$$

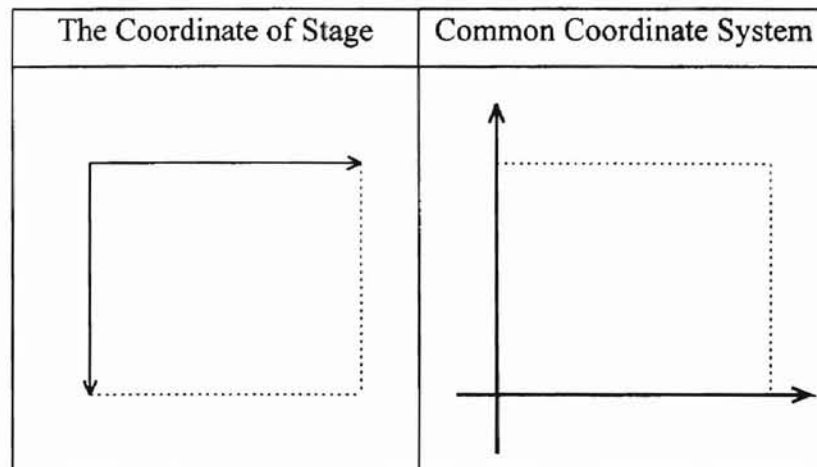


Figure 6-9: The Comparison of Two Coordinate Systems

6.5 P/C SCRIPT AND ANCESTOR SCRIPT

Object-oriented programming is a programming approach that uses objects as its primary logical units for constructing software. Lingo is an OOP language. In Lingo, the parent script is a script from which child objects are derived. A parent script can contain instructions for behaviors common to all objects derived from it. When each child object is created (the “birth” or “new” process), additional instructions can be given concerning the behavior of that individual object. The child object is an entity residing in RAM that was created with reference to a parent script. A child object may control physical objects on the stage as part of its scripting, but is not a physical object in itself. A child object is an occurrence of a parent script. Each child object of the same parent script shares the parent script's handlers but maintains individual values for properties.

A parent script contains three types of Lingo:

- An optional *on new* handler, which creates a new child object and sets its initial values when the handler is called. (The term *me* serves as a local variable that contains the child object itself and provides a placeholder for the child object in Lingo statements.)
- Optional additional handlers that control the child object's behavior and properties after the child object is created.
- An optional statement that declares which variables are property variables--variables for which each child object can maintain individual values regardless of the values for other child objects.

The *new* function (handler) creates a new child object when it uses the name of a parent script. The syntax is *new* (script "scriptName", argument1, argument2, argument3...). The *new* function can be issued from anywhere in the movie or animation and the developer can customize the child object by changing the variable name and values of the arguments in the *new* statement.

The parent-child script is a great way for objects to inherit attributes, but the chain of inheritance can go back still further, to another generation. An ancestor is an additional parent script whose handlers are available to a child object. A parent script makes another parent script its ancestor by assigning the script's name to the ancestor property.

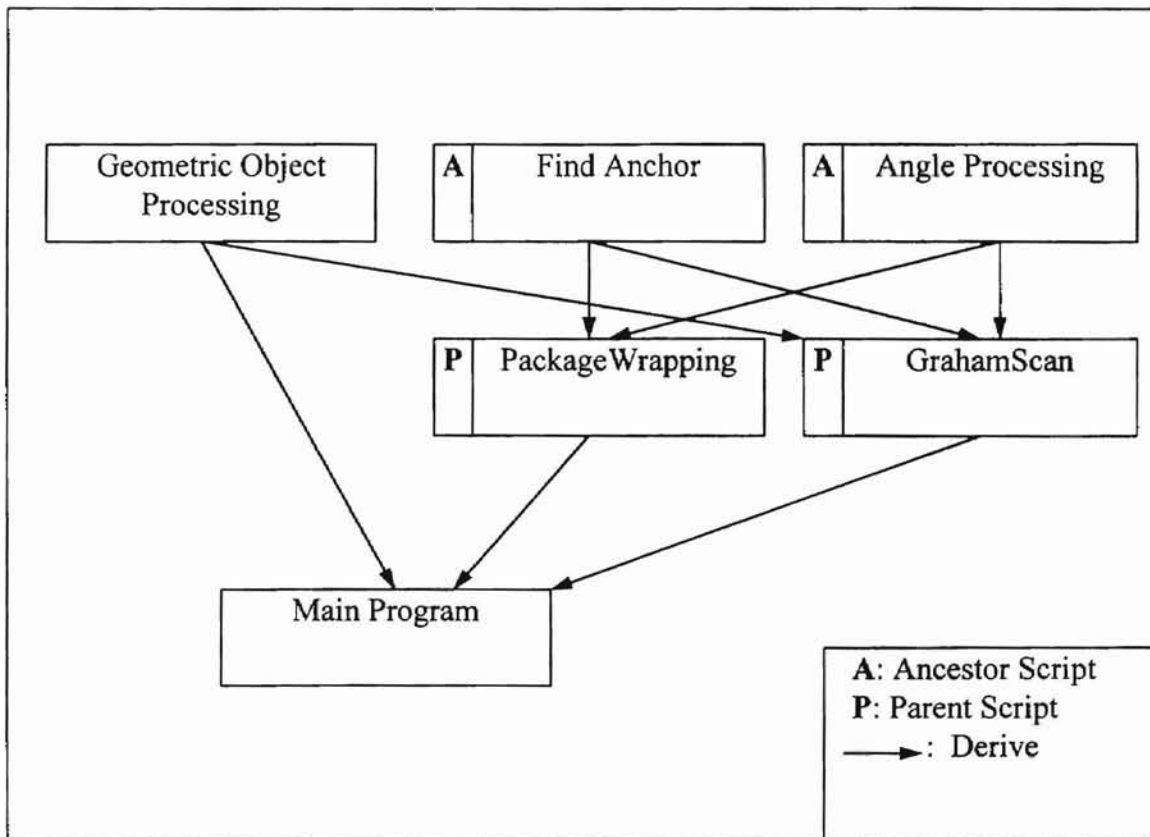


Figure 6-10: The OOP Hierarchy of the System

Lingo is not the only language using the OOP methodology. C++ and Java have many concepts that are under slightly different names with almost the same meanings and functionality: class instead of parent, instance variable instead of property variable, method instead of handler, and super class (or base class) rather than ancestor script.

In the Convex Hull animation system, three levels of P/C script are used, as shown in Figure 6-9. Since PackageWrapping script and GrahamScan script should share the same functions of anchor point searching and angle processing, two-ancestor scripts are created and birthed in PackageWrapping and GrahamScan. Geometric Object Processing has many functions globally can be used by every other script; therefore its seems to be a parent script for PackageWrapping script and GrahamScan script.

6.6 SHOCKWAVE MOVIE GENERATION

To make a movie play from a page on the web, some necessary tags are needed to add to an HTML document. To run a Director movie from an HTML document, requires the EMBED or OBJECT tags. EMBED is the original tag defined by Netscape Navigator. OBJECT is the newer Microsoft Internet Explorer tag. All Shockwave compatible browsers support EMBED; newer browsers support the added functionality of OBJECT. To make sure a movie plays on as many compatible browsers as possible, it is best to use both tags.

Shockwave for Director downloads and plays Director movies in Microsoft Internet Explorer and Netscape Navigator. Millions of web users have downloaded Shockwave from the Macromedia web site; others have received it with their operating system or web browser.

When Director creates a Shockwave movie, it compresses the movie's data to the smallest possible size for downloading. When a user views a web page that includes a Shockwave movie, the data from the movie begins downloading. Shockwave decompresses the data and plays the movie in the user's web browser. If the movie was created for streaming, it begins playing as soon as the data for the first frame has arrived.

CHAPTER VII

SUMMARY, CONCLUSION, AND SUGGESTED FUTURE WORK

7.1 SUMMARY AND CONCLUSION

Geometric animation is a new field in the area of computer science. The boom in computer software and hardware development has caused more excellent tools to appear. These tools greatly benefit the development of geometric animation. The Convex Hull animation system employs Macromedia Director 6 and obtains a great benefit from multimedia presentations and its Web solution, Shockwave movie.

The disadvantage of many multimedia platforms including Macromedia Director is that no built-in function is provided for direct manipulation of geometric objects, such as lines and points. The Convex Hull animation system provides many functions, such as DrawLine, DeleteLine, DrawPoint and DeletePoint, to directly control these geometric objects. This creates a foundation for geometric animation employing multimedia tools like Macromedia Director. Any other geometric animation system developers may load script containing these functions into their Lingo script and use them to manipulate geometric objects.

Some advantages of the Convex Hull animation system come from the OOP features owned by Lingo scripting language. The entire system is implemented by three levels of inherent architecture. This architecture makes it possible to greatly decrease the redundancy and bugs. It also makes it possible to add a new system component without successively modifying the others. This advantage will benefit future system extension

and further development.

The graphical user interface provides a flexible and efficient interaction between user and multimedia animation presentation. Users can interactively control the entire animation and even input the data that they want through the web browser when a Shockwave movie is playing.

7.2 SUGGESTED FUTURE WORK

Since the Convex Hull animation system is designed and implemented under object-oriented methodology, both system structure and functionality can be extended or modified without triggering a chain reaction of program modification. Future extension is possible based on this system.

The Convex Hull Animation System is based on 2D space, which may be extended to 3D in the future. A 3D Convex Hull problem is similar to a 2D problem, that is, searching a Convex Hull over 3D space. Since Macromedia Company has provided the new tool *Extreme 3D2* for 3D animation, the 3D extension would be realistic.

Many other geometric algorithms based on line and point operations can be implemented by extending the Convex Hull animation system.

SELECTED BIBLIOGRAPHY

- [1] Adams, M.L., "Tutorial and Visual Display of the Basic Concepts of Two Dimensional Linear Programming", M.S. Thesis, Oklahoma State University, 1973.
- [2] Amorin, R.V., and Rezende, P.J., "Algorithm Understanding through Dedicated Animation Environments", Technical Report, DCC-25/93, State University of Campinas, Brazil. 1993.
- [3] Baecker, R.M., "Sorting Out Sorting" (video), *Siggraph Video Review*, 7. 1981.
- [4] Brown, M.H., and Sedgewick, R., "A System for Algorithm Animation", *ACM Computer Graphics* (July). 18 (3), pp. 177-186. 1984.
- [5] Brown, M.H., "Algorithm Animation", Ph.D. Thesis, Brown University. 1988.
- [6] Brown, M.H., "Color and Sound in Algorithm Animation", *Proceedings 91 IEEE Workshop on Visual Languages*, pp. 10-17. 1991.
- [7] Brown, M.H., "Zeus: A System for Algorithm Animation and Multi-View Editing", *Proceedings 1991 IEEE Workshop on Visual Languages*. pp. 4-9. 1991.
- [8] Brown, M.H., and Najork, M.A., "Algorithm Animation Using 3D Interactive Graphics", Technical Report 110a, Digital Equipment Corp. Systems Research Center. 1993.
- [9] Burtnyk, N. and Wein, M., "Computer-generated Key-frame Animation", *Journal SMPTE*, Vol. 80, pp. 149-153. 1971.
- [10] DiGiano, C.J., "Visualizing Program Behavior Using Non-speech Audio", M.S. Thesis, University of Toronto. 1992.
- [11] Epstein, P., Kavanagh, J., Knight, A., May, J., Nguyen, T., and Sack, Jörg-Rüdiger, "A Workbench for Computational Geometry", *Algorithmica*, 11(4): 404-428. 1994.

- [12] Graham, R.L., "An efficient algorithm for determining the convex hull of a finite planar set", *Inf. Process. Lett.* 1, pp. 132-133. 1972.
- [13] Harvick, L.H., "Rule Based Data Structure Animation", M.S. Thesis, Oklahoma State University, 1996.
- [14] Hausner, A., "Introduction to the Animation of Geometric Algorithms", <http://www.cs.princeton.edu/~ah/>. 1996.
- [15] Kochanek, D., and Bartels, R., "Interpolating Splines with Local Tension, Continuity and Bias Tension", Proc. SIGGRAPH' 84. *Computer Graphics*. Vol. 18, No. 3, pp. 33-41. 1984.
- [16] Lin, B.H., "An Object-Oriented Graphic User Interface for Visualization of B-Tree's Animator", M.S. Thesis, Oklahoma State University, 1997.
- [17] London, R.L., and Duisberg, R.A. "Animating Programs Using Smalltalk", *IEEE Computer Graphics*. (August) pp. 61-71. 1985.
- [18] Moret, B.M.E., and Shapiro, H.D., *Algorithms from P to NP* (Volume I), Redwood City, CA: Benjamin/Cummings. 1991.
- [19] Pandit, A.V., "Simulation and APL Description of the PDP 11/40", M.S. Thesis, Oklahoma State University, 1975.
- [20] Price, B.A., Baecker, R.M., and Small, I.S., "A Principled Taxonomy of Software Visualization", *Journal of Visual Languages and Computing* 4 (3): 32-39. 1993.
- [21] Reynolds, C.W., "Computer Animation with Scripts and Actors", Proc. SIGGRAPH' 82. pp. 289-296. 1982.
- [22] Rezende, P.J., and Jacometti, W.R., "Geolab: An Environment for Development of Algorithms in Computational Geometry", Technical Report, DCC-26/93, State

- University of Campinas, Brazil. 1993.
- [23] Roberts, J., *Director 6 Demystified*, Berkeley, CA: Macromedia Press. 1998.
- [24] Schmitz, Joe, *Macromedia Director6 – Learning Lingo*, Berkeley, CA: Macromedia Press. 1997.
- [25] Schorn, P., “The XYZ GeoBench for the Experimental Evaluation of Geometric Algorithms”, *Computational Support for Discrete Mathematics*. 1992.
- [26] Shen, B., “Animation of Binary Search Tree”, M.S. Thesis, Oklahoma State University, 1997.
- [27] Smith, N.E., *Intranet Client-Server Applications Using The Web*, Plano, TX: Wordware Publishing, Inc. 1998
- [28] Tal, A.Y., and Dobkin, D.P., “Visualization of Geometric Algorithms”, *IEEE Transactions on Visualization and Computer Graphics (TVCG)*. Volume 1, Number 2. 1995.
- [29] Thalmann, N., and Thalmann, D., “The Use of High Level Graphical Types in the MIRA Animation System”, *IEEE Computer Graphics and Applications*. Vol. 3, No. 9, pp. 98-16. 1983 [30]
- [30] Thalmann, N., and Thalmann, D., *Computer Animation*. Tokyo: Springer-Verlag. 1990.
- [31] Thalmann, N., and Thalmann, D., *Interactive Computer Animation*. New Jersey: Prentice Hall. 1996.
- [32] Thompson, G.R., “Simulation Study of Snoopy Cache Coherence Protocols”, B.S. Thesis, Oklahoma State University, 1996.

- [33] Vaughan, T., Multimedia making it works, Berkeley, CA: Osborne McGraw-Hill. 1996.
- [34] Xu, C., "Animation of Stack, Queue and List", M.S. Thesis, Oklahoma State Univeristy, 1997.
- [35] Öz, O., "Visualization of Learning in Neural Networks", M.S. Thesis, Oklahoma State University, 1997.

VITA

CHAOPANG FAN

Candidate for the Degree of

Master of Science

Thesis: A MULTIMEDIA ANIMATED COMPARISON OF CONVEX HULL
ALGORITHM

Major Filed: Computer Science

Biographical:

Personal Data: Born in Zhenzhou, Henan, P.R. China, June 8, 1968, the son of
Mr. ShangXian Fan and Mrs. Kexue Wu.

Education: Graduated from Guangzhou No. 30 High School, Guangzhou,
Guangdong, P.R. China, in July 1986; received Bachelor of Science
degree in Applied Physics from Shantou University, Shantou, Guandong,
P.R. China in July 1990; Completed requirements for Master of Science
degree at Oklahoma State University in July 1998.

Professional Experience: Computer Engineer, Computer and Information Center,
Guangzhou Iron & Steel Group, 1990-1995. Research Assistantship,
Geology Department, Oklahoma State University, 1997-1998.